compact1, compact2, compact3
java.net

Class Socket

java.lang.Object
  java.net.Socket

All Implemented Interfaces:

`Closeable`, `AutoCloseable`

已知直接子类:

`SSLSocket`

```
public class Socket
extends Object
implements Closeable
```

这个类实现了客户端套接字（也被称为“套接字”）。套接字是两台机器之间的通信的一个端点。

套接字的实际工作是由该类的一个实例进行`SocketImpl`。一个应用程序，通过改变创建套接字实现的套接字工厂，可以配置自己创建适合本地防火墙的套接字。

从以下版本开始:

`JDK1.0`

另请参见:

`setSocketImplFactory(java.net.SocketImplFactory)`, `SocketImpl`, `SocketChannel`

构造方法摘要

构造方法

Modifier	Constructor and Description
	Socket() 创建一个连接的套接字，与socketimpl系统默认的类型。
	Socket(**InetAddress** address, int port) 创建一个流套接字，并将其与指定的IP地址中的指定端口号连接起来。
	Socket(**InetAddress** host, int port, boolean stream) 过时的。 使用UDP传输DatagramSocket。
	Socket(**InetAddress** address, int port, **InetAddress** localAddr, int localPort) 创建一个套接字，并将其与指定的远程端口上的指定的远程地址连接起来。
	Socket(**Proxy** proxy) 创建一个连接的套接字类型，指定代理，如果有，应该使用无论任何其他设置。
protected	**Socket**(**SocketImpl** impl) 创建一个用户指定的socketimpl连接插座。
	Socket(**String** host, int port) 创建一个流套接字，并将其与指定的主机上的指定端口号连接起来。
	Socket(**String** host, int port, boolean stream) 过时的。

使用UDP传输DatagramSocket。

Socket(**String** host, int port, **InetAddress** localAddr, int localPort)
创建一个套接字，并将其连接到指定的远程端口上的指定的远程主机上。

方法摘要

所有方法 **静态方法** **接口方法** **具体的方法**

Modifier and Type	Method and Description
void	**bind**(**SocketAddress** bindpoint) 将套接字绑定到本地地址。
void	**close**() 关闭这个套接字。
void	**connect**(**SocketAddress** endpoint) 将此套接字连接到服务器。
void	**connect**(**SocketAddress** endpoint, int timeout) 将此套接字与指定的超时值连接到服务器。
SocketChannel	**getChannel**() 返回与此套接字关联的独特的 **SocketChannel**对象，如果任何。
InetAddress	**getInetAddress**() 返回套接字连接的地址。
InputStream	**getInputStream**() 返回此套接字的输入流。
boolean	**getKeepAlive**() 如果 **SO_KEEPALIVE**启用。
InetAddress	**getLocalAddress**() 获取绑定的套接字的本地地址。
int	**getLocalPort**() 返回此套接字绑定的本地端口号。
SocketAddress	**getLocalSocketAddress**() 返回此套接字绑定到的端点的地址。
boolean	**getOOBInline**() 如果 **SO_OOBINLINE**启用。
OutputStream	**getOutputStream**() 返回此套接字的输出流。
int	**getPort**() 返回此套接字连接的远程端口号。
int	**getReceiveBufferSize**() 得到这个 Socket的 **SO_RCVBUF**选项的值，是由平台用于该 Socket输入缓冲区的大小。
SocketAddress	**getRemoteSocketAddress**() 返回此套接字连接的端点的地址，或如果它是无关的 null。
boolean	**getReuseAddress**() 如果 **SO_REUSEADDR**启用。
int	**getSendBufferSize**() 得到这个 Socket的 **SO_SNDBUF**期权价值，即缓冲区的大小由平台用于输出在这 Socket。
int	**getSoLinger**() 返回设置 **SO_LINGER**。
int	**getSoTimeout**()

	返回设置 **SO_TIMEOUT**。
boolean	**getTcpNoDelay**() 如果 **TCP_NODELAY**启用。
int	**getTrafficClass**() 获取从这个套接字发送的数据包的IP头中的业务类或服务类型
boolean	**isBound**() 返回套接字的绑定状态。
boolean	**isClosed**() 返回套接字的关闭状态。
boolean	**isConnected**() 返回套接字的连接状态。
boolean	**isInputShutdown**() 返回套接字连接的读半是否关闭。
boolean	**isOutputShutdown**() 返回套接字连接的写是否关闭的是否关闭。
void	**sendUrgentData**(int data) 在套接字上发送一个字节的紧急数据。
void	**setKeepAlive**(boolean on) 启用/禁用 **SO_KEEPALIVE**。
void	**setOOBInline**(boolean on) 启用/禁用 **SO_OOBINLINE**（TCP紧急数据收据）默认情况下，此选项是禁用TCP套接字上接收紧急数是默默丢弃。
void	**setPerformancePreferences**(int connectionTime, int latency, int bandwidth) 设置此套接字的性能首选项。
void	**setReceiveBufferSize**(int size) 集 **SO_RCVBUF**选项，这 Socket指定值。
void	**setReuseAddress**(boolean on) 启用/禁用 **SO_REUSEADDR**套接字选项。
void	**setSendBufferSize**(int size) 设置这个 Socket指定值的 **SO_SNDBUF**选项。
static void	**setSocketImplFactory**(**SocketImplFactory** fac) 设置客户端套接字实现工厂的应用程序。
void	**setSoLinger**(boolean on, int linger) 启用/禁用 **SO_LINGER**与指定的逗留的时间秒。
void	**setSoTimeout**(int timeout) 启用/禁用 **SO_TIMEOUT**以指定的超时时间，以毫秒为单位。
void	**setTcpNoDelay**(boolean on) 启用/禁用 **TCP_NODELAY**（禁用/启用Nagle的算法）。
void	**setTrafficClass**(int tc) 集交通类或从该套接字发送数据包的IP报头字节型服务。
void	**shutdownInput**() 将此套接字的输入流放在“流结束”中。
void	**shutdownOutput**() 禁用此套接字的输出流。
String	**toString**() 将这一 String插座。

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Socket

```
public Socket()
```

创建一个连接的套接字，与socketimpl系统默认的类型。

从以下版本开始:

JDK1.1

Socket

```
public Socket(Proxy proxy)
```

创建一个连接的套接字类型，指定代理，如果有，应该使用无论任何其他设置。

如果存在安全管理器，它的checkConnect方法调用代理主机地址和端口号作为参数。这可能导致SecurityException。

实例：

- `Socket s = new Socket(Proxy.NO_PROXY);`将创建一个普通插座忽略任何其他代理配置。
- `Socket s = new Socket(new Proxy(Proxy.Type.SOCKS, new InetSocketAddress ("socks.mydom.com", 1080)));`将创建一个套接字通过指定的SOCKS代理服务器连接。

参数

`proxy` - `Proxy`对象指定应该使用什么样的代理。

异常

`IllegalArgumentException` -如果代理是无效的类型或 `null`。

`SecurityException` -如果一个安全管理存在和允许连接到代理被拒绝。

从以下版本开始:

一点五

另请参见:

`ProxySelector`， `Proxy`

Socket

```
protected Socket(SocketImpl impl)
          throws SocketException
```

创建一个用户指定的socketimpl连接插座。

参数

`impl` -一个 **`socketimpl`**子类需要使用插座上的一个实例。

异常

`SocketException` -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

JDK1.1

Socket

```
public Socket(String host,
```

```
                int port)
        throws UnknownHostException,
                IOException
```

创建一个流套接字，并将其与指定的主机上的指定端口号连接起来。

如果指定的主机null是指定的地址InetAddress.getByName(null)等效。换句话说，它相当于指定的环回接口地址。

如果应用程序已指定服务器套接字工厂，那个工厂的createSocketImpl方法来创建实际的套接字实现。否则创建一个“普通”套接字。

如果存在安全管理器，它的checkConnect方法被称为主机地址和port作为它的参数。这可能导致SecurityException。

参数

host -主机名称，或 null为环回地址。

port -端口号。

异常

UnknownHostException如果主机的IP地址不能被确定。

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许操作。

IllegalArgumentException如果端口外部端口的值指定的范围内，这是0和65535之间，包容。

另请参见:

setSocketImplFactory(java.net.SocketImplFactory)， SocketImpl，
SocketImplFactory.createSocketImpl()， SecurityManager.checkConnect(java.lang.String, int)

Socket

```
public Socket(InetAddress address,
                int port)
        throws IOException
```

创建一个流套接字，并将其与指定的IP地址中的指定端口号连接起来。

如果应用程序已指定套接字工厂，那个工厂的createSocketImpl方法来创建实际的套接字实现。否则创建一个“普通”套接字。

如果存在安全管理器，它的checkConnect方法被称为主机地址和port作为它的参数。这可能导致SecurityException。

参数

address - IP地址。

port -端口号。

异常

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许操作。

IllegalArgumentException如果端口外部端口的值指定的范围内，这是0和65535之间，包容。

NullPointerException -如果 address是空的。

另请参见:

setSocketImplFactory(java.net.SocketImplFactory)， SocketImpl，
SocketImplFactory.createSocketImpl()， SecurityManager.checkConnect(java.lang.String, int)

Socket

```
public Socket(String host,
                int port,
```

```
                InetAddress localAddr,
                int localPort)
        throws IOException
```

创建一个套接字，并将其连接到指定的远程端口上的指定的远程主机上。插座也将（）到本地地址和端口提供。

如果指定的主机null是指定的地址InetAddress.getByName(null)等效。换句话说，它相当于指定的环回接口地址。

一个zero本地端口号会让系统接自由港在bind操作。

如果存在安全管理器，它的checkConnect方法被称为主机地址和port作为它的参数。这可能导致SecurityException。

参数

host -远程主机的名称，或 null为环回地址。

port -远程端口

localAddr -本地地址的套接字绑定，或 null为 anyLocal地址。

localPort -本地端口的套接字绑定到一个系统，或 zero选择自由港。

异常

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许连接到目的地，或如果它的 checkListen方法不允许绑定到本地端口。

IllegalArgumentException如果端口或localport参数超出指定的有效端口值范围，即0和65535之间，包容。

从以下版本开始：

JDK1.1

另请参见：

SecurityManager.checkConnect(java.lang.String, int)

Socket

```
public Socket(InetAddress address,
                int port,
                InetAddress localAddr,
                int localPort)
        throws IOException
```

创建一个套接字，并将其与指定的远程端口上的指定的远程地址连接起来。插座也将（）到本地地址和端口提供。

如果指定的本地地址null是指定的地址作为anylocal地址等（见InetAddress.isAnyLocalAddress()）。

一个zero本地端口号会让系统接自由港在bind操作。

如果存在安全管理器，它的checkConnect方法被称为主机地址和port作为它的参数。这可能导致SecurityException。

参数

address -远程地址

port -远程端口

localAddr -本地地址的套接字绑定，或 null为 anyLocal地址。

localPort -本地端口的套接字绑定到一个或 zero系统选择自由港。

异常

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许连接到目的地，或如果它的 checkListen方法不允许绑定到本地端口。

IllegalArgumentException如果端口或localport参数超出指定的有效端口值范围，即0和65535之间，包容。

NullPointerException -如果 address是空的。

从以下版本开始:

JDK1.1

另请参见:

SecurityManager.checkConnect(java.lang.String, int)

Socket

```
@Deprecated
public Socket(String host,
                          int port,
                          boolean stream)
                   throws IOException
```

过时的。 *使用UDP传输DatagramSocket。*
创建一个流套接字，并将其与指定的主机上的指定端口号连接起来。

如果指定的主机null是指定的地址InetAddress.getByName(null)等效。换句话说，它相当于指定的环回接口地址。

如果流的说法是true，这将创建一个流式套接字。如果流的说法是false，它创建一个数据报套接字。

如果应用程序已指定服务器套接字工厂，那个工厂的createSocketImpl方法来创建实际的套接字实现。否则创建一个“普通”套接字。

如果存在安全管理器，它的checkConnect方法被称为主机地址和port作为它的参数。这可能导致SecurityException。

如果一个UDP套接字的使用，TCP/IP相关的套接字选项不适用。

参数

host -主机名称，或 null为环回地址。

port -端口号。

stream - boolean指示这是一个流式套接字、数据报套接字。

异常

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许操作。

IllegalArgumentException如果端口外部端口的值指定的范围内，这是0和65535之间，包容。

另请参见:

setSocketImplFactory(java.net.SocketImplFactory)， SocketImpl，
SocketImplFactory.createSocketImpl()， SecurityManager.checkConnect(java.lang.String, int)

Socket

```
@Deprecated
public Socket(InetAddress host,
                          int port,
                          boolean stream)
                   throws IOException
```

过时的。 *使用UDP传输DatagramSocket。*
创建一个套接字，并将其与指定的IP地址中的指定端口号连接起来。

如果流的说法是true，这将创建一个流式套接字。如果流的说法是false，它创建一个数据报套接字。

如果应用程序已指定服务器套接字工厂，那个工厂的createSocketImpl方法来创建实际的套接字实现。否则创建一个“普通”套接字。

如果存在安全管理器，它的checkConnect方法被称为host.getHostAddress()和port作为它的参数。这可能导致

SecurityException。

如果UDP套接字的使用，TCP/IP相关的套接字选项不适用。

参数

host - IP地址。

port -端口号。

stream -如果 true，创建流套接字；否则，创建一个数据报套接字。

异常

IOException -如果一个I / O错误创建套接字时。

SecurityException -如果存在一个安全管理及其 checkConnect方法不允许操作。

IllegalArgumentException如果端口外部端口的值指定的范围内，这是0和65535之间，包容。

NullPointerException -如果 host是空的。

另请参见:

setSocketImplFactory(java.net.SocketImplFactory)， SocketImpl，
SocketImplFactory.createSocketImpl()， SecurityManager.checkConnect(java.lang.String, int)

方法详细信息

connect

```
public void connect(SocketAddress endpoint)
             throws IOException
```

将此套接字连接到服务器。

参数

endpoint - SocketAddress

异常

IOException如果连接时发生错误

IllegalBlockingModeException -如果这插座有对应的通道和通道处于非阻塞模式

IllegalArgumentException如果终点是空的或是通过这个插座不支持SocketAddress类

从以下版本开始:

一点四

connect

```
public void connect(SocketAddress endpoint,
                    int timeout)
             throws IOException
```

将此套接字与指定的超时值连接到服务器。一个零的超时被解释为一个无限超时。该连接将被阻塞，直到建立或发生错误。

参数

endpoint - SocketAddress

timeout -用于毫秒超时值。

异常

IOException如果连接时发生错误

如果连接超时到期之前 SocketTimeoutException

IllegalBlockingModeException -如果这插座有对应的通道和通道处于非阻塞模式

IllegalArgumentException如果终点是空的或是通过这个插座不支持SocketAddress类

从以下版本开始:

一点四

bind

```
public void bind(SocketAddress bindpoint)
          throws IOException
```

将套接字绑定到本地地址。

如果地址是null，然后系统会拿起一个临时端口和一个有效的本地地址绑定套接字。

参数

bindpoint - SocketAddress绑定

异常

IOException如果绑定操作失败，或者如果插座已绑定。

IllegalArgumentException -如果bindpoint是这个插座不支持SocketAddress类

SecurityException -如果存在一个安全管理及其 checkListen方法不允许绑定到本地端口。

从以下版本开始:

一点四

另请参见:

isBound()

getInetAddress

```
public InetAddress getInetAddress()
```

返回套接字连接的地址。

如果插座连接到closed之前，那么这个方法之后会继续关闭套接字连接返回的地址。

结果

远程IP地址此套接字连接，或 null如果套接字没有连接。

getLocalAddress

```
public InetAddress getLocalAddress()
```

获取绑定的套接字的本地地址。

如果存在安全管理器，它的checkConnect方法被调用的本地地址和-1作为它的参数看看操作是允许的。如果操作是不允许的，loopback返回的地址。

结果

本地地址的套接字绑定，环回地址如果由安全经理否认，或通配符地址如果套接字关闭或不绑定但。

从以下版本开始:

JDK1.1

另请参见:

SecurityManager.checkConnect(java.lang.String, int)

getPort

```
public int getPort()
```

返回此套接字连接的远程端口号。

如果插座连接到closed之前，那么这个方法之后会继续关闭套接字连接的端口号返回。

结果

此套接字连接的远程端口号，或0如果套接字未连接。

getLocalPort

```
public int getLocalPort()
```

返回此套接字绑定的本地端口号。

如果插座会被closed之前，那么这个方法之后会继续关闭套接字返回本地端口号。

结果

此套接字绑定或- 1的本地端口号，如果套接字没有绑定到。

getRemoteSocketAddress

```
public SocketAddress getRemoteSocketAddress()
```

返回此套接字连接的端点的地址，或如果它是无关的 null。

如果插座连接到closed之前，那么这个方法之后会继续关闭套接字连接返回的地址。

结果

一个 SocketAddress表示此套接字的远程端点，或 null如果尚未连接。

从以下版本开始：

一点四

另请参见：

getInetAddress(), getPort(), connect(SocketAddress, int), connect(SocketAddress)

getLocalSocketAddress

```
public SocketAddress getLocalSocketAddress()
```

返回此套接字绑定到的端点的地址。

如果一个套接字绑定到一个InetSocketAddress 代表端点是closed，那么这个方法之后会继续关闭套接字返回一个InetSocketAddress。在这种情况下，返回的地址是InetSocketAddress wildcard地址和端口是绑定到本地端口。

如果存在安全管理器，它的checkConnect方法被调用的本地地址和-1作为它的参数看看操作是允许的。如果操作是不允许的，一个SocketAddress代表loopback地址，本地端口套接字绑定返回。

结果

一个 SocketAddress表示此套接字的本地端点，或 SocketAddress代表回送地址如果由安全经理否认，或 null如果插座不绑定但。

从以下版本开始：

一点四

另请参见：

getLocalAddress(), getLocalPort(), bind(SocketAddress), SecurityManager.checkConnect(java.lang.String, int)

getChannel

```
public SocketChannel getChannel()
```

返回唯一 SocketChannel对象与接口相关的，如果任何。

插座将有一个通道如果，只有如果，渠道本身是通过创建SocketChannel.open或ServerSocketChannel.accept方法。

结果

这个插座相关的套接字通道，或 null如果这个插座不是一个通道

从以下版本开始:

一点四

getInputStream

```
public InputStream getInputStream()
                           throws IOException
```

返回此套接字的输入流。

如果这个套接字有一个相关的信道，那么产生的输入流将其所有的操作都委托给信道。如果信道是非阻塞模式，然后输入流的read操作会抛出一个IllegalBlockingModeException。

在异常情况下潜在的连接可能被远程主机或网络软件断（例如在TCP连接的情况下连接复位）。当断开连接时，由网络软件检测到以下应用于返回的输入流：—

- 网络软件可以丢弃由套接字缓冲的字节数。字节不是由网络软件可以读取使用read丢弃。
- 如果没有字节缓冲的插座上，或所有的缓冲字节已被read，那么随后调用read将抛出一个IOException。
- 如果没有字节缓冲的插座，与插座没有关闭使用close，然后available将返回0。

关闭返回InputStream将密切相关的插座。

结果

用于从该套接字读取字节的输入流。

异常

IOException如果I/O错误时创建的输入流时，关闭套接字，该套接字没有连接，或插座输入已关机使用 shutdownInput()

getOutputStream

```
public OutputStream getOutputStream()
                            throws IOException
```

返回此套接字的输出流。

如果这个套接字有一个相关的信道，那么产生的输出流将其所有的操作都委托给信道。如果信道是非阻塞模式，然后输出流的write操作会抛出一个IllegalBlockingModeException。

关闭返回OutputStream将密切相关的插座。

结果

用于写入字节到该套接字的输出流。

异常

IOException如果I/O错误发生时创建输出流或者套接字没有连接。

setTcpNoDelay

```
public void setTcpNoDelay(boolean on)
                   throws SocketException
```

启用/禁用 TCP_NODELAY（禁用/启用Nagle的算法）。

参数

on - true使tcp_nodelay， false禁用。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

JDK1.1

另请参见:

getTcpNoDelay()

getTcpNoDelay

```
public boolean getTcpNoDelay()
                      throws SocketException
```

如果 TCP_NODELAY启用。

结果

一个 boolean指示是否启用 TCP_NODELAY。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

JDK1.1

另请参见:

setTcpNoDelay(boolean)

setSoLinger

```
public void setSoLinger(boolean on,
                        int linger)
                 throws SocketException
```

启用/禁用 SO_LINGER与指定的逗留的时间秒。最大超时值是特定于平台的。设置只影响套接字关闭。

参数

on是否流连忘返。

linger多长时间徘徊，如果是真的。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

IllegalArgumentException如果灵儿值是负的。

从以下版本开始:

JDK1.1

另请参见:

getSoLinger()

getSoLinger

```
public int getSoLinger()
                throws SocketException
```

返回设置 SO_LINGER。-1返回意味着该选项被禁用。设置只影响套接字关闭。

结果

对于 SO_LINGER设置。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始：

JDK1.1

另请参见：

setSoLinger(boolean, int)

sendUrgentData

```
public void sendUrgentData(int data)
                    throws IOException
```

在套接字上发送一个字节的紧急数据。要发送的字节是数据参数的最低八位。紧急字节后发送的任何先前写入插座OutputStream和任何未来的写入输出流之前。

参数

data -数据发送的字节

异常

IOException -如果有错误发送数据。

从以下版本开始：

一点四

setOOBInline

```
public void setOOBInline(boolean on)
                  throws SocketException
```

启用/禁用 SO_OOBINLINE（TCP紧急数据收据）默认情况下，此选项是禁用TCP套接字上接收紧急数据是默默丢弃。如果用户希望接收紧急数据，则必须启用此选项。当启用时，紧急数据被接收到与正常数据的内联。

请注意，只有有限的支持，提供处理传入的紧急数据。特别是，没有传入的紧急数据的通知，也没有能力区分正常的数据和紧急数据，除非提供了一个更高级别的协议。

参数

on - true使 SO_OOBINLINE， false禁用。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始：

一点四

另请参见：

getOOBInline()

getOOBInline

```
public boolean getOOBInline()
                    throws SocketException
```

如果 SO_OOBINLINE启用。

结果

一个 boolean指示是否启用 SO_OOBINLINEis。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点四

另请参见:

setOOBInline(boolean)

setSoTimeout

```
public void setSoTimeout(int timeout)
                  throws SocketException
```

启用/禁用 SO_TIMEOUT以指定的超时时间，以毫秒为单位。这个选项设置为非零超时，在这个插座相关的一read() InputStream电话只为这个时间块。如果超时时间已到，一 **java.net.sockettimeoutexception**抬起，虽然插座仍然有效。选择 必须可以进入阻塞操作有影响之前。超时值必须 > 0。一个零的超时被解释为一个无限超时。

参数

timeout -指定的超时时间，以毫秒为单位。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

JDK 1.1

另请参见:

getSoTimeout()

getSoTimeout

```
public int getSoTimeout()
                 throws SocketException
```

返回设置 SO_TIMEOUT。0返回意味着该选项被禁用（即，超时的无穷大）。

结果

对于 SO_TIMEOUT设置

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

JDK1.1

另请参见:

setSoTimeout(int)

setSendBufferSize

```
public void setSendBufferSize(int size)
                       throws SocketException
```

集 SO_SNDBUF选项，这 Socket指定值。的 SO_SNDBUF选项使用的平台的网络代码的大小设置为底层网络I/O缓冲器的暗示。

因为SO_SNDBUF是一个暗示，那要验证的缓冲设置什么尺寸的程序应该调用getSendBufferSize()。

参数

size的大小来设置发送缓冲区的大小。此值必须大于0。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

IllegalArgumentException -如果该值为0或为负。

从以下版本开始:

一点二

另请参见:

getSendBufferSize()

getSendBufferSize

```
public int getSendBufferSize()
                       throws SocketException
```

得到这个 Socket的 SO_SNDBUF期权价值，即缓冲区的大小由平台用于输出在这 Socket。

结果

为此 Socket的 SO_SNDBUF期权的价值。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点二

另请参见:

setSendBufferSize(int)

setReceiveBufferSize

```
public void setReceiveBufferSize(int size)
                           throws SocketException
```

集 SO_RCVBUF选项，这 Socket指定值。的 SO_RCVBUF选项使用的平台的网络代码的大小设置为底层网络I/O缓冲器的暗示。

增加接收缓冲区的大小可以增加网络I / O的高容量连接的性能，同时降低它可以帮助减少积压的传入的数据。

因为SO_RCVBUF是一个暗示，那要验证的缓冲设置什么尺寸的程序应该调用getReceiveBufferSize()。

对SO_RCVBUF价值也用来设置TCP接收窗口，这样远程对等。一般情况下，当套接字连接时，窗口大小可以被修改。然而，如果一个接收窗口大于64K的要求则必须要求之前插座连接到远程节点。有两种情况要注意：

1. 在接受来自ServerSocket的插座，这必须通过调用ServerSocket.setReceiveBufferSize(int)在ServerSocket绑定到一个地方address.

   做
2. 客户端套接字，setreceivebuffersize()必须连接插座的远程节点前调用。

参数

size的大小来设置接收缓冲区的大小。此值必须大于0。

异常

IllegalArgumentException -如果该值为0或为负。

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点二

另请参见:

getReceiveBufferSize(), ServerSocket.setReceiveBufferSize(int)

getReceiveBufferSize

```
public int getReceiveBufferSize()
                          throws SocketException
```

得到这个 Socket的 SO_RCVBUF选项的值，是由平台用于该 Socket输入缓冲区的大小。

结果

为此 Socket的 SO_RCVBUF期权的价值。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点二

另请参见:

setReceiveBufferSize(int)

setKeepAlive

```
public void setKeepAlive(boolean on)
                  throws SocketException
```

启用/禁用 SO_KEEPALIVE。

参数

on是否有插座保持打开。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点三

另请参见:

getKeepAlive()

getKeepAlive

```
public boolean getKeepAlive()
                     throws SocketException
```

如果 SO_KEEPALIVE启用。

结果

一个 boolean指示是否启用 SO_KEEPALIVE。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点三

另请参见:

setKeepAlive(boolean)

setTrafficClass

```
public void setTrafficClass(int tc)
                     throws SocketException
```

集交通类或从该套接字发送数据包的IP报头字节型服务。由于底层的网络实现可能会忽略这个值应用程序应该考虑它一个提示。

TC 必须范围是`0 <= tc <= 255`或时会抛出。

笔记:

互联网协议的V4的价值由`integer`，其中最重要的8位代表套接字发送的IP数据包的TOS字节的值。RFC 1349定义的TOS值如下：

- `IPTOS_LOWCOST (0x02)`
- `IPTOS_RELIABILITY (0x04)`
- `IPTOS_THROUGHPUT (0x08)`
- `IPTOS_LOWDELAY (0x10)`

最后的低阶位总是忽略这对应于MBZ（必须为零）点。

在优先领域设定位可能导致线程说明操作是不允许的。

RFC 1122节4.2.4.2表明，一个标准的TCP实现，但不是必须让应用程序生命周期的连接在TOS字段的变化。所以无论是服务领域的类型可以在TCP连接已经建立的改变取决于底层平台的实现。应用程序不应假设他们可以在连接变化的TOS字段。

互联网协议V6 `tc`是价值，将被放置到IP报头的sin6_flowinfo场。

参数

`tc - bitset的 int价值。`

异常

`SocketException` -如果有一个错误的交通类或服务类型设置

从以下版本开始:

一点四

另请参见:

`getTrafficClass()`, `SocketOptions.IP_TOS`

getTrafficClass

```
public int getTrafficClass()
                    throws SocketException
```

获取交通类或类型的服务在IP头从插座

作为底层网络实现可以忽略交通类或服务类型设置使用`setTrafficClass(int)`这种方法可能返回比以前使用的方法不同`setTrafficClass(int)`此套接字上的价值。

发送的数据包

结果

已设置的业务类或服务类型

异常

`SocketException` -如果有错误获得交通类或服务类型值。

从以下版本开始:

一点四

另请参见:

`setTrafficClass(int)`, `SocketOptions.IP_TOS`

setReuseAddress

```
public void setReuseAddress(boolean on)
                     throws SocketException
```

启用/禁用 SO_REUSEADDR套接字选项。

当一个TCP连接被关闭的连接可能处于超时状态一段时间后，连接被关闭（通常称为TIME_WAIT 2MSL状态或等待状态）。使用一个众所周知的套接字地址或端口可能无法将套接字绑定到所需的SocketAddress如果有连接超时状态涉及socket地址或端口的应用。

使SO_REUSEADDR绑定套接字使用bind(SocketAddress)允许将套接字绑定即使以前的连接在超时之前的状态。

当一个Socket建立SO_REUSEADDR初始设置是禁用的。

行为时，SO_REUSEADDR启用或禁用后一个套接字绑定（见isBound()）没有定义。

参数

on -是否启用或禁用套接字选项

异常

SocketException -如果出现错误启用或禁用的 SO_REUSEADDR套接字选项，或关闭套接字。

从以下版本开始:

一点四

另请参见:

getReuseAddress(), bind(SocketAddress), isClosed(), isBound()

getReuseAddress

```
public boolean getReuseAddress()
                        throws SocketException
```

如果 SO_REUSEADDR启用。

结果

一个 boolean指示是否启用 SO_REUSEADDR。

异常

SocketException -如果在底层协议有错误，例如TCP错误。

从以下版本开始:

一点四

另请参见:

setReuseAddress(boolean)

close

```
public void close()
           throws IOException
```

关闭这个套接字。

目前在任何线程阻塞I/O操作对该套接字将SocketException。

一旦一个套接字已关闭，它不提供进一步的网络使用（即不能连接或反弹）。需要创建一个新的套接字。

关闭此插座也将关闭套接字的InputStream和OutputStream。

如果这个套接字有一个相关的通道，那么这个通道也关闭了。

Specified by:

close 接口 Closeable

Specified by:

close 接口 AutoCloseable

异常

IOException如果I/O错误发生时关闭此套接字。

另请参见:

isClosed()

shutdownInput

```
public void shutdownInput()
                   throws IOException
```

将此套接字的输入流放在“流结束”中。将发送到套接字的输入流侧的任何数据都被确认，然后默默地丢弃。

如果你阅读后从套接字输入流套接字上调用这个方法，流的available方法将返回0，其read方法将返回-1（结束流）。

异常

IOException如果I/O错误发生时关闭套接字。

从以下版本开始:

一点三

另请参见:

shutdownOutput(), close(), setSoLinger(boolean, int), isInputShutdown()

shutdownOutput

```
public void shutdownOutput()
                    throws IOException
```

禁用此套接字的输出流。对于一个TCP套接字，先前写入的数据将被发送后正常关闭TCP连接。如果你向一个socket写入输出流后调用shutdownoutput()插座上，流将抛出一个异常。

异常

IOException如果I/O错误发生时关闭套接字。

从以下版本开始:

一点三

另请参见:

shutdownInput(), close(), setSoLinger(boolean, int), isOutputShutdown()

toString

```
public String toString()
```

将这一 String插座。

重写:

toString 方法重写，继承类 Object

结果

这个套接字的字符串表示。

isConnected

```
public boolean isConnected()
```

返回套接字的连接状态。

注意：关闭套接字不清楚其连接状态，这意味着这种方法将返回true为封闭式插座（见isClosed()）如果成功连接被关闭之

前。

结果

如果插座成功连接到服务器

从以下版本开始:

一点四

isBound

```
public boolean isBound()
```

返回套接字的绑定状态。

注意：关闭套接字不清楚其结合状态，这意味着这种方法将返回`true`为封闭式插座（见`isClosed()`）如果成功会被关闭之前。

结果

如果插座成功绑定到一个地址

从以下版本开始:

一点四

另请参见:

`bind(java.net.SocketAddress)`

isClosed

```
public boolean isClosed()
```

返回套接字的关闭状态。

结果

如果套接字已关闭，则是

从以下版本开始:

一点四

另请参见:

`close()`

isInputShutdown

```
public boolean isInputShutdown()
```

返回套接字连接的读半是否关闭。

结果

如果套接字的输入已关闭，则为

从以下版本开始:

一点四

另请参见:

`shutdownInput()`

isOutputShutdown

```
public boolean isOutputShutdown()
```

返回套接字连接的写是否关闭的是否关闭。

结果

如果插座的输出已关闭，则为

从以下版本开始：

一点四

另请参见：

`shutdownOutput()`

setSocketImplFactory

```
public static void setSocketImplFactory(SocketImplFactory fac)
                                 throws IOException
```

设置客户端套接字实现工厂的应用程序。工厂只能指定一次。

当一个应用程序创建一个新的客户端套接字，该套接字实现工厂的`createSocketImpl`方法来创建实际的套接字实现。

通过`null`的方法是一种无OP除非厂家已经设置。

如果存在安全管理器，该方法首先调用安全管理器的`checkSetFactory`方法确保操作是允许的。这可能导致SecurityException。

参数

`fac` -所需的工厂。

异常

`IOException` -如果一个`I / O`错误设置套接字工厂时。

`SocketException` -如果工厂已经定义。

`SecurityException` -如果存在一个安全管理及其 `checkSetFactory`方法不允许操作。

另请参见：

`SocketImplFactory.createSocketImpl()`， `SecurityManager.checkSetFactory()`

setPerformancePreferences

```
public void setPerformancePreferences(int connectionTime,
                                       int latency,
                                       int bandwidth)
```

设置此套接字的性能首选项。

插座使用TCP / IP协议默认。一些实现可能提供替代协议比TCP / IP不同的性能特点。这种方法允许应用程序来表达自己的喜好，如何将这些权衡时，应作出实施选择从可用的协议。

性能偏好描述由三个整数，其值表明的相对重要性的短连接时间，低延迟，和高带宽。整数的绝对值是不相关的，为了选择一个协议的值进行简单的比较，具有较大的值，表示更强的偏好。否定的值代表一个较低的优先级比积极的价值观。如果应用程序更喜欢短连接时间在低延迟和高带宽，例如，就可以调用这个方法的价值`(1, 0, 0)`。如果应用更高的带宽比低延迟，低延迟上述连接时间短，就可以调用这个方法的价值`(0, 1, 2)`。

此方法调用此方法后，已连接将不会有任何效果。

参数

`connectionTime - int`表达短连接时间的相对重要性

`latency - int`表达低延迟的相对重要性

`bandwidth - int`表达高带宽的相对重要性

从以下版本开始：

一点五

Submit a bug or feature
For further API reference and developer documentation, see Java SE Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

本帮助文档是使用《百度翻译》翻译，请与英文版配合使用 by--QQ:654638585